Answer Set Programming's Contributions to Classical Logic

An analysis of ASP methodology

Marc Denecker, Joost Vennekens, Hanne Vlaeminck, Johan Wittocx, and Maurice Bruynooghe

Department of Computer Science, K.U. Leuven

Abstract. Much research in logic programming and non-monotonic reasoning originates from dissatisfaction with classical logic as a knowledge representation language, and with classical deduction as a mode for automated reasoning. Discarding these classical roots has generated many interesting and fruitful ideas. However, to ensure the lasting impact of the results that have been achieved, it is important that they should not remain disconnected from their classical roots. Ultimately, a clear picture should emerge of what the achievements of answer set programming mean in the context of classical logic, so that they may be given their proper place in the canon of science. In this paper, a look at different aspects of ASP, in an effort to identify precisely the limitations of classical logic that they exposed and investigate how the ASP approaches can be transferred back to the classical setting. Among the issues we thus address are the closed world assumption, "classical" and default negation, default reasoning with exceptions, definitions, lp-functions and the interpolation technique and the strong introspection operator. We investigate the ASP-methodology to encode knowledge using these language constructs and come across a dichotomy in the ASP-methodology.

1 Introduction

In the late fifties, a thesis of logic-based AI was that classical first order logic (FO) and deductive theorem provers for FO would play a key role in building intelligent systems [21]. In the late sixties and early seventies, the disappointing results of this project led a number of new research directions. In logic-based AI, two areas that arose as a reaction were logic programming (LP) and non-monotonic reasoning. In logic programming, the idea was that FO was too expressive and had to be restricted to Horn clause logic to allow efficient theorem proving [16]. The origin of non-monotonic reasoning lies in the dissatisfaction about the use of FO for representing common sense knowledge. To overcome the limitations of FO, new logics and logic principles were studied. Examples are the Closed World Assumption [26], Circumscription [22] and Default logic [28]. Although having completely different points of departure, both areas converged in the late eighties and early nineties mainly under the influence of Michael Gelfond

and Vladimir Lifschitz who sought to explain logic programming with negation as failure using non-monotonic reasoning principles [12], and turn it into an expressive knowledge representation language [13]. Michael and Vladimir thus became founding fathers of the field of Answer Set Programming (ASP) [1], which has developed into a flourishing, dynamic field with many applications and powerful systems [9, 8].

Currently, there is undeniably a large conceptual gap between ASP and classical logic. As a long-term situation, this is undesirable. First of all, despite its limitations, FO provides a set of essential connectives, together with a well-understood methodology for their use, which makes it an invaluable kernel for KR languages. Second, the conceptual gap also hampers recognition of ASP's contributions to AI and computational logic, thus compromising the use of the ASP languages and tools in other fields.

This paper will try to identify limitations of classical logic exposed by Gelfond's work and investigate how the ASP solutions can be transferred back into classical logic. This boils down to identifying important forms of knowledge, to study how they are represented in ASP and to investigate how they can be represented in (extensions of) FO. Thus, this paper also studies the methodology of ASP, and compares it with the methodology of FO. It identifies also key contributions of the ASP language and integrates them with FO. Among the issues we thus address are the closed world assumption, "classical" and default negation, default reasoning with exceptions, definitions, lp-functions and the interpolation technique and the strong introspection operator.

Integrating classical logic and ASP is not a trivial task, as research on rule languages for the semantic web is now also discovering [25]. Expanding the stable semantics to the syntax of FO as in [10] is obviously *redefining* FO, not *integrating* ASP and FO. Among subtle problems and tricky mismatches, one glaring difference clearly stands out: ASP uses the *answer set* as its basic semantic construct, whereas FO of course uses *interpretations* (or *structures*). There is a profound difference between the two concepts. As defined in [13], an answer set is a set of literals that represents a possible state of belief that a rational agent might derive from the logic program. In other words, an answer set is *subjective*; it does not talk directly about the world itself, but only about the beliefs of an agent about this world. This conception of answer sets is not a philosophical afterthought, but is tightly connected with the view of negation as failure as a modal, epistemic or default operator, and therefore truly lies at the heart of ASP. By contrast, FO has a *possible world* semantics, in which each model of a theory represents a state of the world that is possible according to this theory. In other words, FO is concerned with *objective* forms of knowledge, that do not involve propositional attitudes such as the belief or knowledge of an epistemic agent, etc. Modal extensions of classical logic that can express such forms of subjective knowledge typically use *Kripke structures* (or the special case of *sets of possible worlds*) as a semantic object.

Answer sets fall between classical interpretations and Kripke structures: an answer set represents a belief state, and thus differs from an interpretation which

describes a possible state of the world, but it does so in a less complex and less accurate way than a Kripke structure or sets of possible worlds, since it only describes what literals are believed rather than arbitrary formulas. This mismatch between ASP's basic semantic constructs and the semantic constructs of FO and its modal extensions seriously complicates the task of formally comparing ASP with FO, and it will be the focus of much of this paper. We will come across a dichotomy in the use of ASP programs: in some, an answer set represents the belief state of an existing rational agent, in many others the programmer carefully crafts the mind of an imaginary agent so that solutions of a problem can be extracted from the belief sets of that agent. This dichotomy stands orthogonal to the (many) ways in which stable semantics can be formalised [18]. We will show that this leads to two different methodologies and, de facto, to two different KR languages, as pointed out long time ago by Gelfond and Lifschitz[13] and, from a different angle, in [6].

Overview In Section 2 we briefly recall the relevant basics of answer set programming and first order logic. In Section 3, we discuss the basic semantic principles of knowledge representation in ASP and contrast them with those of FO. We distinguish two methodologies for knowledge representation in ASP. In Section 4 we introduce FO(ID), the extension of first order logic with inductive definitions. In Section 5 we take a close look at different forms of objective knowledge, how they are represented in ASP and what is their representation in FO(ID). We discuss unique names and domain closure assumptions, different forms of closed world assumptions, the representation of definitions in ASP and defaults. In Section 6, we extend FO with a simple epistemic operator with which we can simulate negation as failure and strong negation.

2 Preliminaries of ASP and FO

We briefly recall the basic concepts of Answer Set Programming and of classical logic. A vocabulary Σ consists of a set Σ_{Fun} of constant and function symbols and a set Σ_{Pred} of predicate symbols. An answer set program P over vocabulary Σ is a collection of rules $\mathtt{l} \mathtt{:-} \mathtt{l_1}, \ldots, \mathtt{l_m}, \mathtt{not}\ \mathtt{l_{m+1}}, \ldots, \mathtt{not}\ \mathtt{l_n}.$ and constraints: $\mathtt{:-}\ \mathtt{l_1}, \ldots, \mathtt{l_m}, \mathtt{not}\ \mathtt{l_{m+1}}, \ldots, \mathtt{not}\ \mathtt{l_n}.$ where each $\mathtt{l}, \mathtt{l_i}$ is an atom $\mathtt{P}(\mathtt{t_1}, \ldots, \mathtt{t_k})$ or a strong negation literal $\neg \mathtt{P}(\mathtt{t_1}, \ldots, \mathtt{t_k})$. A constraint is seen as a special rule defining $\mathtt{F}$ as a new symbol: $\mathtt{F} \mathtt{:-}\ \mathtt{not}\ \mathtt{F}, \mathtt{l_1}, \ldots, \mathtt{l_m}, \mathtt{not}\ \mathtt{l_{m+1}}, \ldots, \mathtt{not}\ \mathtt{l_n}$. An answer set program with variables is seen usually as the set of all ground instances obtained by substituting ground terms of Σ for all variables. A disjunctive answer set program is a set of rules where the head l may be a disjunction $\mathtt{l'_1}\ \mathtt{v}\ \ldots\ \mathtt{v}\ \mathtt{l'_k}$ of atoms and strong negation literals.

An answer set A of P is a set of ground atoms and strong negation literals such that A is a $\subseteq$-minimal element in the collection of all sets S of such literals that have the property that, for each rule

$$\mathtt{l'_1 v \ldots v l'_k} \mathtt{:-} \mathtt{l_1}, \ldots, \mathtt{l_m}, \mathtt{not}\ \mathtt{l_{m+1}}, \ldots, \mathtt{not}\ \mathtt{l_n}.$$

if $l_1, \ldots, l_m \in S$ and $l_{m+1}, \ldots, l_n \notin A$, then at least one $l'_i \in S$.

Note that for a constraint, there exists at least one literal l_i such that either $1 \leq i \leq m$ and $l_i \notin A$ or $m+1 \leq i \leq n$ and $l_i \in A$.

As for the informal semantics, an answer set program P is seen as the representation of all knowledge of some rational introspective epistemic agent. A rule $\mathtt{l}'_1\mathtt{v}\ldots\mathtt{v}\mathtt{l}'_\mathtt{k}\mathtt{:-}\,\mathtt{l}_1,\ldots,\mathtt{l}_\mathtt{m},\mathtt{notl}_{\mathtt{m}+1},\ldots,\mathtt{notl}_\mathtt{n}.$ expresses that this agent believes one of $\mathtt{l}'_\mathtt{k}$ if he believes $\mathtt{l}_1,\ldots,\mathtt{l}_\mathtt{m}$ to be true and considers it possible that each of $\mathtt{l}_{\mathtt{m}+1},\ldots,\mathtt{l}_\mathtt{n}$ is false. The agent obeys a *rationality principle*, that is that he does not believe an atom or a strong negation literal unless his theory gives him an argument for believing it. In the answer set semantics, this rationality principle is formalized by the minimality requirement. Each answer set represents the set of believed literals in one of the possible states of belief of the agent.

First order logic (FO) formulas φ over vocabulary Σ are constructed using standard inductive rules for $\wedge, \neg, \vee, \Rightarrow, \Leftrightarrow, \exists, \forall$. A Σ-interpretation I (or Σ-structure) consists of a domain U_I, for each function symbol F/n an n-ary function $F^I : U_I^n \to U_I$ and for each predicate symbol P/n an n-ary relation P^I on U_I. The interpretation of a term and the satisfaction of a formula under I and some variable assignment θ is defined by standard inductive rules, such as $(F(t_1,\ldots,t_n))^{I\theta} = F^{I\theta}(t_1^{I\theta},\ldots,t_n^{I\theta})$ and the (non-monotonic) inductive rule for the truth of negated formulas: $I\theta \models \neg\varphi$ if $I\theta \not\models \varphi$. The interpretation of variable-free ground terms t and sentences φ does not depend on the variable assignment θ. We use t^I to denote t's interpretation and write $I \models \varphi$ (I is a *model*) to denote that φ is satisfied in $I\theta$ for arbitrary θ.

A Herbrand interpretation is one in which U_I is the set of ground terms of Σ (called the Herbrand universe) and I interprets each ground term t by itself, i.e., for each function symbol F and all term tuples $t_1,\ldots,t_n$, $F^I(t_1,\ldots,t_n) = F(t_1,\ldots,t_n)$. An alternative representation of a Herbrand interpretation is as a set of its true ground atoms. In this representation, a Herbrand interpretation is mathematically identical to an answer set of Σ. However, this is accidental and should not be taken to mean that both concepts are the same: in Tarskian model semantics, an interpretation (Herbrand or otherwise) does not represent a state of belief of an agent.

3 Basic principles of knowledge representation in ASP and FO

An exquisite property of FO is the clarity and precision of the informal semantics of its connectives and quantifiers, and consequently, of its formulas. Tarskian model semantics is a mathematical theory developed to formalize this informal semantics. In the Tarskian model semantics, a model of an FO theory T is viewed as an abstract representation of a state of the world that satisfies all axioms in T, hence that is a *possible world* according to T. This makes FO an *objective* language in that the truth of its formulas can be evaluated in the context of an objective world state. This is in contrast to negation-as-failure literals `not P` in

ASP rules, whose truth is not determined by the material world but by the state of belief of an epistemic agent.

An FO theory T over Σ can of course also be understood as representing a state of belief of some agent. The belief state of T is formalized by the collection[1] of its models: a model is a state of the world that is *possible* in that state of belief. Such a collection of interpretations, henceforth called a *collection (or set, in case) of possible worlds*, offers an extremely detailed representation of a belief state. Therefore, sets of possible worlds are used in the semantics of (propositional) epistemic logics such as kd45 as the representation of the belief state of the agent; such sets correspond to Kripke structures with the total accessibility relation.

As a representation of a belief state of an agent, a collection of possible worlds is vastly more precise than a (single) answer set. From any set of possible worlds, the corresponding answer set is the set of literals that are satisfied in all. But vice versa, not much information can be derived from an answer set of an agent about his collection of possible worlds. For instance, take $\Sigma = \{P, Q\}$ and assume that all we know about some agent is that his answer set is $\emptyset$, i.e., he or she knows (can derive) no literals. There are many sets of possible worlds with that answer set; the agents possible world set could be the set of models of the tautology *true*, or of $P \vee Q$, or $\neg P \vee Q$, or $P \vee \neg Q$, or $\neg P \vee \neg Q$, or $P \Leftrightarrow Q$ and more. In each case, he would not be able to derive a literal. Knowing only this answer set, all we can tell about the possible worlds is that there is at least one in which P is true and one in which P is false, and the same for Q. There is not a single interpretation that can be decided to be a possible world or rejected to be one. Hence, an answer set is a very coarse representation of a belief state. As we illustrate now, this has major impact on ASP methodology.

An answer set of an ASP program represents the belief of a rational agent—but who is this agent? The rational agent could be an existing epistemic entity in the real world: it could be a knowledge base that is being queried, or it could be us, the people writing the ASP program. As an example of this, we take the following scenario from [13].

Example 1. Assume a database with complete knowledge about the GPA (Grade Point Average) of students and partial knowledge about whether they belong to a minority group. A student is eligible for a grant if his GPA is high, or if he has a fair GPA and belongs to a minority. The policy of the department is to interview students who may be (but are not necessarily) eligible. In ASP, incomplete databases can be represented by a theory consisting of ground literals, such as:

$$\texttt{FairGPA(Bob)}.\ \ \texttt{Minority(Bob)}.\ \ \neg\texttt{Minority(Dave)}.$$
$$\texttt{FairGPA(Ann)}.\ \ \texttt{HighGPA(John)}.$$

The partial completeness of this database is expressed by two CWA rules:

$$\neg\texttt{FairGPA(x)} \leftarrow \texttt{not FairGPA(x)}.$$
$$\neg\texttt{HighGPA(x)} \leftarrow \texttt{not HighGPA(x)}.$$

[1] In general, the collection of models of an FO theory is too large to be a set.

The next rule expresses who is to be interviewed:

$$\texttt{Interview(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{not Minority(x)}, \texttt{not}\neg\texttt{Minority(x)}.$$

In the database, Ann is the only student with fair GPA and unknown minority status. This ASP program has a unique answer set including the literals in the database and the atom `Interview(Ann)`. This corresponds to our beliefs about this domain; hence, the rational agent can be viewed to be us, the ASP programmer. Our possible worlds are all Herbrand interpretations satisfying the literals in the answer set. In the CWA rules and the `Interview` rule, the epistemic nature of negation as failure is well exploited, in the latter to verify that the KB knows that person x is neither a minority nor a non-minority. This is an interesting sort of application that cannot be directly expressed in classical logic.

Many present-day examples of ASP concern computational problems such as graph coloring, Hamiltonian path, planning and scheduling problems, and many others. The only epistemic entity around in such domains is us, the people writing the ASP programs. Our knowledge is about data and the properties of a correct solution. The solutions that we want our systems to compute are given by certain output predicates in worlds that are possible according to this knowledge – the coloring predicate in a state in which the graph is correctly colored, the action predicates in a state where the actions form a correct plan, and so on. Unfortunately, our knowledge in such domains is highly disjunctive and there is no way in which possible worlds could be reconstructed from the set of literals that we "believe", i.e., that can be derived from our knowledge. For such domains, ASP developed a successful alternative methodology that makes use of a "virtual" non-existent agent. The "mind" of this figment of our imagination has been craftily constructed by us, the answer set programmer, in such a way that each of his possible belief sets corresponds to a single correct solution of the problem, i.e., to a *possible world*.

Example 2. The following ASP program represents the graph coloring problem for a given graph:

```
color(x, R) v color(x, B) v color(x, G) :- node(x).
:- color(x,c), color(y,c), edge(x,y).
node(1).  node(2). node(3). edge(1,2). edge(2,3). edge(1,3).
```

This program produces 6 answer sets, each of which corresponds to a coloring of the graph. E.g., the answer set

```
{ node(1).  node(2). node(3). edge(1,2). edge(2,3). edge(1,3).
  color(1,R). color(2,B). color(3,G).}
```

tells us that the (unique) color of 1 is R, that of 2 is B, and that of 3 is G. Obviously, this one answer set does not describe *our* beliefs about the colorings of this graph, since we know that there also exists, e.g., a coloring in which node 1 is not R.

In this example, and in many other ASP applications, *our* knowledge is represented by the *set of possible worlds* consisting of all Herbrand interpretations that are mathematically identical to the answer sets of the virtual agent. And there is no way in which the possible worlds, and hence, the solutions to the problem, could be extracted from *our* belief set.

The above observations show us that in ASP two different methodologies are in use. That of Example 1, called henceforth the ASP-belief methodology, is to represent the belief state(s) of an existing epistemic agent in the field. The second, called ASP-world methodology (Example 2), aims to characterize a collection of possible worlds by an answer set program of a virtual agent whose states of belief correspond to possible worlds. There is a simple heuristic criterion to determine which of these methodologies is used in an ASP application. Indeed, a real epistemic agent such as us or an existing knowledge base may have incomplete knowledge about the application domain, but in practice it is clear what this agent believes. I.e., the epistemic agent has a unique state of belief. Hence, ASP-belief programs typically have a unique answer set while ASP-world programs often have multiple answer sets.

The distinction between the ASP-belief and ASP-world methodologies is reminiscent of a discussion in Michael Gelfond's and Vladimir Lifschitz's seminal 1991 paper [13]. In this paper, they state that a set of rules without strong negation and disjunction can be viewed either as a *general logic program* or as an answer set program. In both cases the semantics of such a rule set is determined by the same stable models. The crucial difference, dixit Gelfond and Lifschitz, is that a stable model of the rule set, viewed as a general logic program, represents a Herbrand interpretation, i.e., a possible world, while it represents an answer set in the second case. There is an evident match here between general logic programming and the ASP-world methodology: ASP-world is the KR methodology of general logic programming. The distinction between general logic programs and answer set programs lays purely on the informal level. This might explain why it did not enter the eye of the ASP community and why the concept of general logic programs has not catched on in the ASP community. Yet, at present, it seems that the vast majority of ASP applications follows the ASP-world methodology and these programs are therefore to be viewed as (extensions of) general logic programs.

The relevance of distinguishing between these two methodologies is that in each different style, different kinds of knowledge can be represented, or the same knowledge is to be represented in different ways. Which methodology is chosen depends on the available knowledge and on the problem to be solved, e.g., whether a solution is (part of) a possible world, or what a KB knows/believes. If we want to match ASP to FO, our effort will therefore need to be parameterized on the particular way in which ASP is used. In the remainder of this paper, we will investigate how different sorts of knowledge can be represented in ASP-belief or ASP-world and in (extensions of) FO.

4 Adding (inductive) definitions to FO

A prototypical ASP-world application is the Hamiltonian cycle problem. The goal is to compute a Hamiltonian cycle through the graph, i.e., a path that passes through each node exactly once and then returns to the initial node. A typical ASP encoding is given by:

```
in(X, Y) v out(X, Y) :- edge(X,Y).
reached(X) :- reached(Y), in(Y,X).
reached(X) :- start(Y), in(Y,X).
:- node(X), not reached(X).
:- in(X,Y), in(X,Z), Y != Z.
:- in(X,Y), in(Z,Y), X != Z.
node(1).  node(2). node(3). edge(1,2). edge(2,3). edge(1,3).
```

The reachability relation defined in this program is a well-known example of an inductively definable concept that cannot be expressed in FO[17]. Definitional knowledge is an important form of human expert knowledge [4, 7]. While simple non-inductive definitions can be expressed through *explicit definitions* $\forall \bar{x}(P(\bar{x}) \Leftrightarrow \varphi_P[\bar{x}])$, FO needs to be extended for inductive definitions. In the logic FO(LFP) [17], FO is extended with a least fixpoint construct. The definition of the reached relation would be expressed as:

$$\forall x(Reached(x) \Leftrightarrow \mathbf{lfp}_{R,x}[(\exists y\ (Start(y) \vee R(y)) \wedge In(y,x))](x)$$

In FO(ID) [7], inductive definitions are represented in a rule-based syntax, similar to the ASP encoding.

$$\left\{ \begin{array}{l} \forall x\ Reached(x) \leftarrow \exists y\ Reached(y) \wedge In(y,x). \\ \forall x\ Reached(x) \leftarrow \exists y\ Start(y) \wedge In(y,x). \end{array} \right\}$$

A rule (over Σ) is an expression of the form $\forall \bar{x}\ P(\bar{t}) \leftarrow \varphi$ where $P(\bar{t})$ is an atomic formula and φ an FO formula, and $\leftarrow$ is the *definitional implication*. A definition Δ is a set of rules. A predicate symbol P in the head of a rule of Δ is called a *defined predicate* of Δ; a symbol of Σ that occurs in Δ but is not defined by it is called a *parameter* or *open symbol* of Δ. The set of defined predicates is denoted $Def(\Delta)$, the remaining symbols $Param(\Delta)$. An FO(ID) theory is a set of FO sentences and definitions.

The satisfaction relation $\models$ of FO(ID) is defined through the standard inductive rules for FO with one additional rule:

- $I \models \Delta$, if $I|_{Def(\Delta)} = \mathrm{wfm}^{\Delta}_{I|_{Param(\Delta)}}$

where $\mathrm{wfm}^{\Delta}_{I|_{Param(\Delta)}}$ is the *parameterized well-founded model* of Δ in the interpretation $I|_{Param(\Delta)}$. That is, to check whether I is a model, we restrict I to the parameter symbols of Δ, extend this interpretation by performing the well-founded model construction in [29][2] and verify whether this well-founded model

[2] The well-founded model construction of [29] extends the original one of [30] by allowing arbitrary FO bodies and parameter symbols (interpreted by an extensional database).

coincides with the (2-valued) interpretation I on the defined symbols. The well-founded semantics formalizes the most common forms of inductive definitions such as monotone inductive definitions (e.g., reachability) and (non-monotone) definitions by induction over a well-founded order (e.g., the satisfaction relation $I \models \varphi$) [7].

In the next section, we identify useful, objective forms of knowledge that can be nicely encoded in ASP and we investigate how to represent them in FO(ID).

5 Representing objective knowledge in ASP and FO(ID)

5.1 Representing UNA and DCA

Implicitly, ASP maintains the *Unique Names Assumption* (UNA(Σ)) [27] that all ground terms of Σ_{Fun} represent different objects, and the *Domain Closure Assumption* (DCA(Σ)) [27] that each object in the universe is represented by at least one ground term of Σ_{Fun}. These assumptions hold in many applications (e.g., databases), but neither UNA nor DCA are imposed in FO. That is, in FO they should be explicitly expressed.

UNA(Σ) is expressed by the FO theory $UNA_{FO}(\Sigma)$ that consists of, for each pair of different function symbols $F/n, G/m \in \Sigma$ [27]:

$$\forall \bar{x}\bar{y}\ \neg(F(\bar{x}) = G(\bar{y}))$$
$$\forall \bar{x}\bar{y}\ F(\bar{x}) = F(\bar{y}) \Rightarrow \bar{x} = \bar{y}$$

It follows from the compactness of FO, that DCA(Σ) cannot be expressed in FO, but it can be expressed in FO(ID) through the following theory $DCA_{FO(ID)}(\Sigma)$:

$$\left\{ \begin{array}{l} \ldots \\ \forall \bar{x}\ U(F(\bar{x})) \leftarrow U(x_1) \wedge \cdots \wedge U(x_n). \\ \ldots \end{array} \right\}$$
$$\forall x\ U(x)$$

where the definition consists of one rule for each constant or function symbol $F/n \in \Sigma$ and U is a new predicate representing the universe. The models of $UNA_{FO}(\Sigma) \wedge DCA_{FO(ID)}(\Sigma)$ are the Herbrand interpretations (up to isomorphism).

In many applications, UNA and DCA are too strong. For instance, in the applications typically considered in description logics, neither holds. In other applications, these axioms are applicable to only a subset σ of Σ_{Fun}. A (very) early example of this is Peano arithmetic, Peano's standard second order theory of the natural numbers where $\sigma = \{0, S/1\}$. This theory contains $UNA_{FO}(\sigma)$ and a second order axiom expressing DCA(σ), i.e., the induction axiom. Thus, UNA and DCA are not applied to $+/2$ and $\times/2$, the two other function symbols of Peano's theory.

In view of this, a number of variants of the ASP language have been designed with relaxations of UNA and/or DCA. An early one in which UNA still holds

but DCA is dropped was proposed by Gelfond and Przymusinska [14]. In [19], an extension of ASP with functions is defined. This ASP extension applies the kind of relaxed UNA+DCA(σ) axioms as observed above in Peano arithemetic. As a consequence, other function symbols represent arbitrary functions in the Herbrand universe of σ. In Open Answer Set Programming [15], UNA and DCA are dropped altogether. Applications of all these extensions arguably follow the ASP-world methodology: answer sets represent belief states of a virtual agent but possible worlds of the human expert.

5.2 Different forms of CWA in complete and partial databases

The general idea of the Closed World Assumption (CWA) [26] in the context of a theory T is the assumption that atoms that cannot be derived from T are false. In the context of incomplete knowledge, this principle often leads to inconsistency and needs to be relaxed. There are different options to do that. In fact, ASP-belief and ASP-world provide *different* forms of CWA.

In ASP-belief, CWA is expressed through ASP formulas of the form:

$$\neg \mathtt{P}(\bar{\mathtt{x}}) \ \mathtt{:-\ not\ P}(\bar{\mathtt{x}}).$$

Informally, the rule states that $\mathtt{P}(\bar{\mathtt{x}})$ is false if it is consistent to assume so. Such rules can be used to represent databases with complete and partial knowledge. Thus, the epistemic `not` connective of ASP allows us to explicitate a local form of CWA.

Example 3. An ASP-belief representation of a complete database :

$$\mathtt{FairGPA(Bob).\ Minority(Bob).\ FairGPA(Ann).\ HighGPA(John).}$$
$$\neg\mathtt{FairGPA(x)} \leftarrow \mathtt{not\ FairGPA(x)}.$$
$$\neg\mathtt{HighGPA(x)} \leftarrow \mathtt{not\ HighGPA(x)}.$$
$$\neg\mathtt{Minority(x)} \leftarrow \mathtt{not\ Minority(x)}.$$

By deleting the CWA rule for `Minority` and adding strong negation literal $\neg$`Minority(Dave)` we obtain the partial database with incomplete knowledge of Example 1.

In ASP-world, no CWA axioms need to be added; on the contrary, axioms are needed to express where the CWA does not hold. Thus in ASP-world, (a form of) CWA is implicit and needs to be overruled when necessary.

Example 4. An ASP-world representation of the complete database of Example 3:

$$\mathtt{FairGPA(Bob).\ Minority(Bob).\ FairGPA(Ann).\ HighGPA(John).}$$

The partial database of Example 1 is represented by extending this with a constraint and a disjunctive rule to open up `Minority`:

$$\mathtt{Minority(x)\ v\ Minority^{*}(x).}$$
$$\mathtt{:-Minority(Dave).}$$

There are several alternative syntaxes to do the latter in ASP, such as the lparse syntax "`0{Minority(x)}1`." or the traditional cycle over negation:

```
Minority(x) :- not Minority*(x).
Minority*(x) :- not Minority(x).
```

Interestingly, the mathematical closure principle underlying stable semantics, according to which an atom belongs to a stable model only if it can be derived by a rule, acts as a rationality principle in ASP-belief, but as a form of CWA in ASP-world! This is a clear example of how the same form of knowledge is represented in different ways in ASP-belief or ASP-world. Note that adding CWA-rules in ASP-world such as $\neg$`FairGPA(x)` $\leftarrow$ `not FairGPA(x)` is of no use: it expresses that the virtual epistemic agent has CWA on this predicate but not that we, the ASP programmer, have CWA on $FairGPA$. The effect is merely to add strong negation literals such as $\neg$`FairGPA(John)` to all answer sets, but strong negation literals are irrelevant since they are ignored when computing the possible worlds from the virtual agent's answer sets.

Recapitulating, in ASP-belief, CWA is not imposed unless explicit rules are added; in ASP-world, CWA is imposed unless overridden explicitly by rules that open up an atom.

The CWA cannot be expressed directly in FO, due to FO's monotonicity. However, a local form of CWA can be simulated through predicate completion [5] , as in the axiom:

$$\forall x(FairGPA(x) \Leftrightarrow x = Bob \vee x = Ann)$$

An alternative solution in FO(ID) is to use definitions to close predicates. The above partial database is represented in FO(ID) as:

$$\begin{array}{l} \{\, FairGPA(Bob).\ \ FairGPA(Ann).\,\} \\ \{\, HighGPA(John).\,\} \\ Minority(Bob).\ \ \neg Minority(Dave). \end{array}$$

Inductive definitions in mathematics quite clearly comprise some form of CWA. Indeed, an object or tuple does not belong to an inductively defined set *unless it is explicitly derived by a rule application.* It is not uncommon to find inductive definitions in mathematical texts that contain explicit CWA-like statements, as in the following definition:

> We define the set of natural numbers by induction:
> - 0 is a natural number;
> - if n is a natural number then $n + 1$ is a natural number;
> - **no other objects are natural numbers.**

Thus, definitions are an informal form of CWA that we understand with great precision. In one of the next sections, we will use this form of CWA to represent defaults.

The following example shows that the CWA's expressed in ASP-belief differ from the one implicit in ASP-world and in definitions.

Example 5. In the context of the complete student database of Example 3, the following ASP-belief rules express which students are eligible for a grant :

```
Eligible(x) :- HighGPA(x).
Eligible :- FairGPA(x), Minority(x).
¬Eligible(x) :- not Eligible(x)
```

There is a problem with this representation in the context of the incomplete database. Indeed, Ann has a fair GPA but an unknown Minority status, and hence, is unknown to be eligible. Hence, the CWA rule jumps to the wrong conclusion that she is not eligible[3]. An ASP-world representation of eligibility is obtained by dropping the CWA rule from this definition. This is similar to FO(ID), where the predicate *Eligible* can be defined as:

$$\left\{ \begin{array}{l} \forall x(Eligible(x) \leftarrow HighGPA(x). \\ \forall x(Eligible \leftarrow FairGPA(x) \wedge Minority(x)). \end{array} \right\}$$

The ASP-world representation as well as the FO(ID) representation is correct whether the database is complete or only partially complete.

An ASP-belief CWA rule turns ignorance of an atom into falsity. Intuitively, if there is one possible world in the agents belief state where the atom is false, the atom is false in all possible worlds. In contrast, in ASP-world and in definitions of FO(ID), a defined atom is false in a possible world if it cannot be produced by rule application in that particular world. Thus, in one world, Ann might be eligible and in another she might not be. This is a weaker form of CWA than the one expressed in ASP-belief, but more useful in this application. This clearly shows that the form of CWA that can be expressed in ASP-belief differs from the form of CWA implicit in ASP-world and in FO(ID) definitions.

5.3 Representing definitions in ASP

Definitions are important in KR [4]. Many applications, also in ASP, contain rules that make up for a definition of one or more predicates. Examples in this paper are the rule defining `Interview` in Example 1 and the rules for `Eligible` in the previous section. (Inductive) definitions can be correctly represented in ASP-world and of course in FO(ID). It is more difficult to represent them in ASP-belief, as we saw in the previous section.

As shown in Example 5, representing definitions in ASP-belief is simple as long as the ASP program has complete knowledge on the parameter predicates. In that case, it suffices to add a CWA rule to the rules expressing the definition. Such a representation unfortunately leads to errors in case of incomplete databases (The parameter predicate `Minority` is incomplete in Example 5). To solve this problem in ASP-belief, we can use the interpolation technique developed by Baral, Gelfond and Kosheleva in [2, 3]. They develop a technique to

[3] We will discuss a solution for this problem in Section 5.3.

approximate *lp-functions* by an answer set program. An lp-function consists of a set of rules (without strong negation) and a set of input and output parameters. There is a clear and intuitive match between definitions of FO(ID) and lp-functions: a definition can be seen as an lp-function with the definitions parameters matching the input parameters, and the defined predicates matching the output parameters of the corresponding lp-function. The algorithm that they propose can be used to translate the rules for *Eligible* into an interpolation: an answer set program that approximates the original rules in the context of an incomplete database. Applying this algorithm on the rules of *Eligible* yields:

$$\begin{aligned}
&\texttt{MayBeMinority(x)} \leftarrow \texttt{not}\neg\texttt{Minority(x)}.\\
&\texttt{Eligible(x)} \leftarrow \texttt{HighGPA(x)}.\\
&\texttt{Eligible(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{Minority(x)}.\\
&\texttt{MaybeEligible(x)} \leftarrow \texttt{HighGPA(x)}.\\
&\texttt{MaybeEligible(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{MayBeMinority(x)}.\\
&\neg\texttt{Eligible(x)} \leftarrow \texttt{notMaybeEligible(x)}.
\end{aligned}$$

The algorithm adds predicates representing students that might be minorities or might be eligible and defines $\neg$`Eligible` as the complement of the *maybe eligible* predicate.

The algorithm depends on which predicates the database has incomplete knowledge about and needs to be modified when more parameter predicates are unknown (e.g., `FairGPA`, `HighGPA`). The correctness and the accuracy of the approach depends on certain conditions as expressed in the correctness theorem in [3]. In particular, if there is disjunctive knowledge on the parameters of the definition, accuracy will be lost. For instance, if we add to the database that also John has a fair GPA and either he or Ann is a minority but not both, then the natural conclusion that either John or Ann is not eligible cannot be drawn from an interpolation.

This shows that representing definitions in ASP-belief has its pitfalls. Also, it shows that introducing incomplete knowledge in ASP-belief has more impact on the representation than in ASP-world and FO(ID).

5.4 Representing defaults

Nonmonotonic Reasoning has its motivation in the problems of FO for representing common sense knowledge [23]. One problem is the *qualification problem*: many universally quantified statements are subject to a very long, if not endless list of qualifications and refinements. The standard example is the statement that *birds can fly*. But what about ostriches? Chickens? Chicks? Birds with clipped or broken wings? Etc. All we can say is: *normally*, a bird can fly, or *most* birds can fly. This gave rise to the domain of *defaults* and *default reasoning*. Defaults are approximative statements and reason about normality (typically, normally, usually, often, most, few, ..).

An essential part of human intelligence is the ability to reason with incomplete, vague, and uncertain information. When such information is represented

in classical logic, where sentences are either wholly true or wholly false, the result is a crisp approximation of the fuzzy reality. The knowledge of a human expert evolves with time, e.g., when he learns more about particular objects or comes across new special cases and exceptions. Hence, the theory has to be continually refined and revised. An important issue here is *elaboration tolerance*; the ease by which new knowledge can be integrated in an old theory.

Let us illustrate this in a traditional example scenario. Assume that we want to represent knowledge about locomotion of animals. We know that most birds fly and that Tweety and Clyde are birds. If that is all we know, it is reasonable to assume that both fly. One approximately correct FO theory that entails the desired conclusions is:

$$T = \{\, \forall x(Bird(x) \Rightarrow Fly(x)), Bird(Tweety) \wedge Bird(Clyde) \,\}$$

Next, we find out that Tweety is a penguin and want to extend our theory to, e.g.,

$$\forall x(Penguin(x) \Rightarrow Bird(x)), \forall x(Penguin \Rightarrow \neg Fly(x)), Penguin(Tweety)$$

However, this theory is inconsistent, and we also need to weaken the axiom that birds fly:

$$\forall x(Bird(x) \wedge \neg Penguin(x) \Rightarrow Fly(x))$$

Unfortunately, we now have lost the desirable but defeasible conclusion that Clyde flies. Adding the reasonable assumption $\neg Penguin(Clyde)$ (since "most birds are not penguins") would solve the problem, at the risk of adding potentially false information, since our assumption may be wrong. And even if Clyde turns out to be an eagle, we might later find out that he is a chick, and these don't fly. Clearly, FO's lack of elaboration tolerance makes it quite messy to maintain our formula that most birds fly.

An important goal of nonmonotonic reasoning is to design *elaboration tolerant* logics and methodologies that support this process of representing and refining evolving expert knowledge. For defaults, ASP achieves elaboration tolerance by means of its non-monotonic negation `not`, which allows additional exceptions to existing rules to be added with minimal effort:

```
Flies(x) :- Bird(x), not Abnormal(x).
Abnormal(x) :- Penguin(x).
```

Adding an additional exception can now be done by the almost trivial operation of introducing a new rule with the abnormality predicate in its head.

In the context of FO, we can achieve a remarkably similar effect, by again turning to the inductive definition construct of FO($\cdot$). Indeed, inductive definitions are essentially a non-monotonic language construct, in the sense that the operation of adding a new definitional rule to an existing definition is a non-monotonic operation, due to the CWA embodied in a definition. We can

leverage this property to represent a default d stating that "Ps are typically Qs" as follows:

$$\forall x\ P(x) \wedge \neg Ab_d(x) \Rightarrow Q(x)$$
$$\{\forall x\ Ab_d(x) \leftarrow \mathbf{f}.\}$$

This theory defines the abnormality predicate as empty, and is therefore equivalent to the sentence $\forall x\ P(x) \Rightarrow Q(x)$. However, when new exceptions to the default come to light, they can easily be added as new definitional rules. For instance, if we discover that the default does not apply to Rs:

$$\forall x\ P(x) \wedge \neg Ab_d(x) \Rightarrow Q(x)$$
$$\left\{\begin{array}{l}\forall x\ Ab_d(x) \leftarrow \mathbf{f}.\\ \forall x\ Ab_d(x) \leftarrow R(x).\end{array}\right\}$$

This has the effect of turning the theory into one equivalent to $\forall x\ P(x) \wedge \neg R(x) \Rightarrow Q(x)$, but does so in an elaboration tolerant way.

In a sense, every definition includes a kind of default information, namely that the defined concept is false by default, where the rules express the exceptions. For instance, the definition:

$$\{\forall x\ Square(x) \leftarrow Rectangle(x) \wedge Rhombus(x).\}$$

expresses that objects typically are not squares, with those objects that are both a rectangle and a rhombus forming an exception to this general rule. This allows us to represent certain kinds of interplaying defaults in a quite compact but still elaboration tolerant way. For instance, the following definition expresses that "animals typically do not fly (d_1), but birds are an exception, in the sense that these typically do fly (d_2), unless they happen to be penguins, which typically do not fly (d_3)":

$$\left\{\begin{array}{l}\forall x\ Flies(x) \leftarrow Bird(x) \wedge \neg Ab_{d_2}(x).\\ \forall x\ Ab_{d_2}(x) \leftarrow Penguin(x) \wedge \neg Ab_{d_3}(x).\\ \forall x\ Ab_{d_3}(x) \leftarrow \mathbf{f}.\end{array}\right\}$$

A new exception to d_1 might be that all bats also fly: this is the simple update of adding the definitional rule $\forall x\ Flies(x) \leftarrow Bat(x) \wedge \neg Ab_{d4}(x)$ and an empty definition for the latter abnormality predicate. A new exception to d_2 might be a bird with a broken wing: $\forall x\ Ab_{d_2}(x) \leftarrow BrokenWing(x)$. Finally, a penguin with a jet pack: $\forall x\ Ab_{d_3}(x) \leftarrow JetPack(x)$.

6 Adding an epistemic operator to FO(·)

In this section, we are concerned with expressing inherently epistemic ASP applications of the kind that can be expressed in ASP-belief. For this, we return to

Example 1, where we had an ASP-belief partial knwoledge base with the following rule to express that students with fair GPA and unknown minority status need to be interviewed:

$$\texttt{Interview(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{not Minority(x)}, \texttt{not}\neg\texttt{Minority(x)}.$$

In ASP-world, the latter rule does not work for the simple reason that the virtual agent has different beliefs about minority students than the knowledge base (or we) has. E.g., in the context of the ASP-world partial database of Example 4, adding the above rule would lead to one answer set containing `Minority(Ann)` and no `Interview(Ann)` and another answer set without `Minority(Ann)` but with `Interview(Ann)`. This is not what we need.

To express this example in a natural way, an epistemic operator is needed as provided in autoepistemic logic (AEL [24]). Applying the translation from ASP to AEL [12], we obtain a correct AEL representation:

$$FairGPA(Bob) \wedge Minority(Bob) \wedge \neg Minority(Dave) \wedge FairGPA(Ann)$$
$$\forall x(\neg FairGPA(x) \Leftarrow \neg K FairGPA(x))$$
$$\forall x(Interview(x) \Leftarrow FairGPA(x) \wedge \neg K Minority(x) \wedge \neg K \neg Minority(x))$$

In general, due to its self-referential nature, an AEL theory may have multiple stable expansions, each describing a different state of belief. This would be highly undesirable in this application, since we want the knowledge base to come up with a single set of people to be interviewed. Fortunately, the above theory is a stratified one. Therefore, it has a unique stable expansion [20] which correctly captures what the knowledge base knows.

For applications of this kind, it will suit us to develop an approach that is both simpler and closer to FO than AEL. Given that we already have an FO theory defining our incomplete database and its associated set of possible worlds, it is natural to consider a layered approach, where we keep our original database theory and just add an additional layer on top of that. This new layer is again just another FO theory, with the exception that it may query the knowledge contained in the database theory. More generally, it is of course equally possible to consider many layers, each building on the knowledge contained in previous layers. This suggests the following straightforward multi-modal extension of FO for reasoning on knowledge bases composed from multiple knowledge sources.

Definition 1. *An ordered epistemic theory $\mathcal{T}$ over vocabulary Σ is a finite set of multi-modal logic theories over Σ that satisfies the following conditions:*

- *Each $T \in \mathcal{T}$ is a multi-modal logic theory, possibly using modal operators $K_{T'}$, where $T' \in \mathcal{T}$.*
- *The modal operators are used in a stratified way, i.e., there is a partial order $<$ on $\mathcal{T}$ and if the modal literal $K_T\varphi$ occurs in a formula of subtheory T' or in the scope of another modal literal $K_{T'}\psi$, then $T < T'$.*

Intuitively, each $T \in \mathcal{T}$ is expressing the knowledge of a component of the knowledge base. Theories in higher levels possess the knowledge of lower theories

and can query lower levels through expressions $K_T\varphi$. Note that if T is minimal in $<$, then it is an FO theory. If T is maximal, then K_T occurs nowhere in $\mathcal{T}$.

Self-referential autoepistemic statements such as $T = \{\neg K_T P \Rightarrow P\}$ cannot be expressed in ordered epistemic logic. The benefit is that FO semantics extends easily. To keep things simple and standard, we consider only Herbrand interpretations of the vocabulary Σ. This effectively means that DCA(Σ) and UNA(Σ) hold[4] and moreover, that all function and constant symbols σ are *rigid*: they have the same interpretation in all possible worlds. Let $\mathcal{T}|_T$ denote the restriction of $\mathcal{T}$ to $\{T' \in \mathcal{T} | T' \leq T\}$.

For any (Herbrand) Σ-interpretation M and variable assignment θ, we define that $M\theta$ satisfies a formula φ, denoted $M\theta \models \varphi$, by extending the standard inductive rules of FO with one extra rule[5]:

- $M\theta \models K_T\varphi$ if for each interpretation M' such that $M' \models \mathcal{T}|_T$, it holds that $M'\theta \models \varphi$.

As usual, we define $M \models \varphi$ for sentences φ -without free variables- if for some variable assignment θ, $M\theta \models \varphi$. M satisfies an ordered epistemic theory $\mathcal{T}$ if it satisfies each sentence in each $T \in \mathcal{T}$. Thus, a theory T can be seen as the distributed knowledge of a layered set of agents, where each higher agent posses the knowledge of its lower agents but not vice versa. Just like an FO theory, an ordered epistemic theory defines a unique possible world set, namely the set of all its models.

A useful feature of this logic is that it is straightforward to extend it with other objective language constructs in FO($\cdot$) such as definitions and aggregates. In the sequel, we will include definitions in the examples.

It is easy to see how our example fits into ordered epistemic logic. Let DB be either our FO or FO(ID) representation—it does not matter which—of the incomplete database. We now build on top of this a second layer $T_I > DB$, consisting of a single definition:

$$T_I = \left\{ \left\{ \begin{array}{l} \forall x\ Interview(x) \leftarrow FairGPA(x) \wedge \\ \qquad\qquad \neg K_{DB}(Minority(x)) \wedge \neg K_{DB}(\neg Minority(x)). \end{array} \right\} \right\}$$

Alternatively, we could of course also have used an FO equivalence:

$$T_I' = \left\{ \begin{array}{l} \forall x\ Interview(x) \Leftrightarrow FairGPA(x) \wedge \\ \qquad\qquad \neg K_{DB}(Minority(x)) \wedge \neg K_{DB}(\neg Minority(x)). \end{array} \right\}$$

Note that we have strengthened the original *implication* specifying $Interview$ into an arguably more accurate *definition*.

[4] A standard way in modal logic of relaxing the assumption of a fixed domain in all worlds is to introduce a unary universe predicate U and allowing only relativized quantifier formulas $\exists x(U(x) \wedge \varphi)$ and $\forall x(U(x) \Rightarrow \varphi)$.

[5] Observe that this is the second time in this paper that we extend FO's satisfaction relation $\models$ by adding a rule to its definition (the first time was for formal FO(ID) definitions). This illustrates that the elaboration tolerant, non-monotonic update operations that we used in Section 5.4 in the FO(ID) encoding of defaults, occurs also in informal inductive definitions.

It is easy to see that the ordered epistemic theory $\mathcal{T} = (DB, T_I)$ has two models: one in which Ann is a minority member, one in which she is not. Therefore, $\mathcal{T}$ entails that only Ann needs an interview.

While in many useful cases, an ordered epistemic theory can be viewed as an autoepistemic theory, we will not delve in this.

Disjunction in ASP-belief– a epistemic operator in ASP-world

As shown in the beginning of this section, negation as failure cannot be used as epistemic operator to query partial ASP-world knowledge bases. In [11], Gelfond investigated this problem in the context of disjunctive answer set programs. To motivate his approach, Gelfond illustrated the epistemic use of negation as failure with Example 1 including the ASP-belief rule:

$$\texttt{Interview(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{not Minority(x)}, \texttt{not } \neg\texttt{Minority(x)}.$$

This rule will correctly identify the students to be interviewed in all simple ASP-belief databases. Next, Gelfond considered a database with the disjunctive information that either Ann or Mike are members of a minority group:

$$\texttt{FairGPA(Ann)}.\ \texttt{FairGPA(Mike)}.$$
$$\texttt{Minority(Ann) v Minority(Mike)}.$$

In this case, we would like to infer that both Ann and Mike need to be interviewed. However, the program has two answer sets, one with `Interview(Ann)`, and another with `Interview(Mike)`, and therefore, the answer of this program to, e.g., the query `Interview(Mike)` is unknown.

To solve this problem, Gelfond proposed to add another epistemic operator K to ASP, called the *strong introspection operator*, that queries whether a literal is present in *all* answer sets. Using this operator, we can correctly express the definition of $Interview$ by:

$$\texttt{Interview(x)} \leftarrow \texttt{FairGPA(x)}, \texttt{not K not Minority(x)}, \texttt{not K Minority(x)}.$$

The resulting program has two correct answer sets:

$$\{\texttt{FairGPA(Ann)}, \texttt{FairGPA(Mike)}, \texttt{Minority(Ann)}, \texttt{Interview(Ann)}, \texttt{Interview(Mike)}\}$$
$$\{\texttt{FairGPA(Ann)}, \texttt{FairGPA(Mike)}, \texttt{Minority(Mike)}, \texttt{Interview(Ann)}, \texttt{Interview(Mike)}\}$$

This representation not only works for Gelfonds disjunctive datatase but also for the ASP-world partial database of Example 4.

In ordered epistemic logic, the problem could be solved by the theory $\mathcal{T}$ consisting of $DB < T_1$ where T_1 is as before and

$$DB = \left\{ \begin{array}{l} FairGPA(Ann).\ FairGPA(Mike.) \\ Minority(Ann) \vee Minority(Mike). \end{array} \right\}$$

This is a correct representation of the scenario and entails that both will be interviewed.

What Gelfond basically observed here is that the use of disjunction in the head turns an ASP-belief program into an ASP-world one. Before adding the disjunction, the answer set represents a belief state of the knowledge base; after adding the disjunction, this is no longer the case. The belief state of the knowledge base is reflected by the set of possible worlds in which either Ann or Mike is a minority. This belief set does not correspond to any of the two answer sets of the disjunctive program. Thus, the use of disjunction in ASP leads here and in many other cases to ASP-world programs. What we can conclude here is as follows:

- Without strong introspection, ASP-world does not provide an autoepistemic operator. ASP-world is not suitable to express introspective statements.
- ASP-belief is not suitable to express disjunctive information.
- Introduction of disjunction (in any of the known ways: disjunctive rule, loop over negation, weight constraints) in a programs including epistemic operators may force us to modify the ASP program. In comparison, the representation in ordered epistemic FO(.) is more elaboration tolerant.

In conclusion, this section has considered an interesting class of epistemic ASP-belief programs, namely, those in which conclusions need to be drawn from incomplete knowledge bases. ASP's combination of strong negation and NAF allows such problems to be elegantly represented and most applications in which strong negation plays a significant role seem to be of this kind. These applications naturally exhibit a layered structure, in which each layer is a knowledge base with its own area of expertise, and different layers may build on each other's knowledge. This suggest the simple epistemic logic that we have presented in this section. This logic has the benefit of simplicity and, moreover, it integrates effortlessly into classical logic or its extensions. While the logic is weaker than AEL, in the sense that it does not allow self reference, it guarantees a unique epistemic model, and it is possible to prove that inference tasks have a lower complexity than in AEL and in disjunctive answer set programming.

7 Conclusion

Gelfond and Lifschitz have merged ideas of logic programming and non-monotonic reasoning to build a practically useful KR-language — answer set programming. In this paper, we have highlighted and analysed these contributions by investigating what they contribute to classical first order logic (FO). We studied how forms of knowledge for which ASP was designed can be represented in FO or in suitable extensions of FO.

A prime factor in our analysis turned out to be the dichotomy between the ASP-belief and ASP-world methodologies: whether a program is written so that an answer set represents a belief set of an existing agent or a possible world. This decision turns out to have an overwhelming effect on the knowledge representation process in ASP:

– Some knowledge principles are implicit in one and not in the other (e.g., CWA).
– Some forms of knowledge can only be expressed in one (e.g., disjunction, autoepistemic statements).
– Some forms of knowledge can be expressed in both, but in considerably different ways (e.g., definitions – with interpolation in ASP-belief).

In the development of ASP, Gelfond and Lifschitz were led by the ASP-belief view. In the current state of the art, it seems that by far most applications of ASP are developed according to ASP-world view. We have explained the reason for this: answer sets are simply too coarse grained to represent the belief state of a real agent or knowledge base in sufficient detail.

To consolidate the contributions of ASP to KR, we believe that the best strategy is to integrate its contributions to FO. The same holds also for other KR-extensions of logic programming such as abductive logic programming. In the past, we have been led by this idea in the development of FO(.). We have shown here how ASP's KR-contributions can be mapped to elaboration tolerant FO(.) theories. For the development of FO(.), we are indebted to Michael Gelfond and Vladimir Lifschitz, whose work, as can be seen in this paper, has been a continous source of inspiration.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
2. C. Baral, M. Gelfond, and O. Kosheleva. Approximating general logic programs. In *ILPS*, pages 181–198, 1993.
3. C. Baral, M. Gelfond, and O. Kosheleva. Expanding queries to incomplete databases by interpolating general logic programs. *J. Log. Program.*, 35(3):195–230, 1998.
4. R. J. Brachman and H. J. Levesque. Competence in knowledge representation. In *National Conference on Artificial Intelligence (AAAI'82)*, pages 189–192, 1982.
5. K. L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.
6. M. Denecker. What's in a model? Epistemological analysis of logic programming. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *KR*, pages 106–113. AAAI Press, 2004.
7. M. Denecker and E. Ternovska. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL)*, 9(2):Article 14, 2008.
8. M. Denecker, J. Vennekens, S. Bond, M. Gebser, and M. Truszczyński. The second Answer Set Programming competition. In Erdem et al. [9], pages 637–654.
9. E. Erdem, F. Lin, and T. Schaub, editors. *Logic Programming and Nonmonotonic Reasoning, 10th International Conference, LPNMR 2009, Potsdam, Germany, September 14-18, 2009. Proceedings*, volume 5753 of *LNCS*. Springer, 2009.
10. P. Ferraris, J. Lee, and V. Lifschitz. A new perspective on stable models. In M. M. Veloso, editor, *IJCAI*, pages 372–379, 2007.
11. M. Gelfond. Strong introspection. In *AAAI*, pages 386–391, 1991.

12. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.
13. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.
14. M. Gelfond and H. Przymusinska. Reasoning on open domains. In *LPNMR*, pages 397–413, 1993.
15. S. Heymans, D. V. Nieuwenborgh, and D. Vermeir. Open answer set programming with guarded programs. *ACM Trans. Comput. Log.*, 9(4), 2008.
16. R. A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.
17. L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
18. V. Lifschitz. Twelve definitions of a stable model. In M. García de la Banda and E. Pontelli, editors, *ICLP*, volume 5366 of *LNCS*, pages 37–51. Springer, 2008.
19. F. Lin and Y. Wang. Answer set programming with functions. In G. Brewka and J. Lang, editors, *KR*, pages 454–465. AAAI Press, 2008.
20. V. W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.
21. J. McCarthy. Programs with common sense. In *Teddington Conference on the Mechanization of Thought Processes*, 1958.
22. J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.
23. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.
24. R. C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984. Reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.
25. J. Pührer, S. Heymans, and T. Eiter. Dealing with inconsistency when combining ontologies and rules using dl-programs. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *ESWC (1)*, volume 6088 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2010.
26. R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, 1977.
27. R. Reiter. Equality and domain closure in first-order databases. *Journal of the ACM*, 27(2):235–249, 1980.
28. R. Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.
29. A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.
30. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.